NEW YORK CITY COLLEGE OF TECHNOLOGY
The City University of New York

DEPARTMENT:	Mathematics
COURSE:	MAT 2540
TITLE:	Discrete Structures and Algorithms II
DESCRIPTION:	This course continues the discussion of discrete mathematical structures and algorithms introduced in MAT 2440. Topics in the second course include predicate logic, recurrence relations, graphs, trees, digital logic, computational complexity and elementary computability.
TEXT:	Discrete Mathematics and its Applications, 8th edition by Kenneth H. Rosen McGraw-Hill
CREDITS:	3 (2 class hours, 2 lab hours)
PREREQUISITES:	MAT 2440

Prepared by Professors Henry Africk, Brad Isaacson, Caner Koca, Nan Li, Satyanand Singh, Arnavaz Taraporevala, Johann Thiel (Fall 2017)
Revised by Professor Johann Thiel (Fall 2020)

A. Testing Guidelines:
   The following exams should be scheduled:
   1. A one-hour exam at the end of the First Quarter
   2. A one-session exam at the end of the Second Quarter
   3. A one-hour exam at the end of the Third Quarter
   4. A one-session Final Examination

B. A Computer Algebra System will be used in class and for a project.

Course Learning Outcomes	General Education Learning Outcomes	Flexible Core- Scientific World
Understand and apply the concept of mathematical induction.	Acquire strategies and skills to create and analyze mathematical proofs using induction. Be able to use appropriate language to communicate mathematical ideas.	Evaluate evidence and arguments critically or analytically. Produce well-reasoned written or oral arguments using evidence to support conclusions.
Identify and solve counting problems.	Be able to solve problems using various counting techniques.	Identify and apply the fundamental concepts and methods of a discipline or interdisciplinary field exploring the scientific world, including, but not limited to: computer science, logic, and mathematics. Produce well-reasoned written or oral arguments using evidence to support conclusions.
Compare data structures and algorithms.	Be able to understand the limitations and implications of an algorithm. Be able to analyze pseudocode to determine the functionality and efficiency of an algorithm. Be able to generate algorithms and effectively communicate their purpose.	Gather, interpret, and assess information from a variety of sources and points of view. Demonstrate how tools of science, mathematics, technology, or formal analysis can be used to analyze problems and develop solutions.
Use the master theorem to solve recurrences that arise from divide-and-conquer algorithms.	Understand the implications and consequences of the master theorem and how it applies to divide-and-conquer algorithms.	Identify and apply the fundamental concepts and methods of a discipline or interdisciplinary field exploring the scientific world, including, but not limited to: computer science, logic, and mathematics. Produce well-reasoned written or oral arguments using evidence to support conclusions.
Study graphs and trees, and solve algorithmic problems such as finding shortest paths or spanning trees.	Acquire a foundation of knowledge of important mathematical concepts and definitions. Be able to see the connections of graph theory to other disciplines and real world problems. Be able to apply various graph-theoretic algorithms.	Understand the scientific principles underlying matters of policy or public concern in which science plays a role.

New York City College of Technology Policy on Academic Integrity

Students and all others who work with information, ideas, texts, images, music, inventions, and other intellectual property owe their audience and sources accuracy and honesty in using, crediting, and citing sources. As a community of intellectual and professional workers, the College recognizes its responsibility for providing instruction in information literacy and academic integrity, offering models of good practice, and responding vigilantly and appropriately to infractions of academic integrity. Accordingly, academic dishonesty is prohibited in The City University of New York and at New York City College of Technology and is punishable by penalties, including failing grades, suspension, and expulsion. The complete text of the College policy on Academic Integrity may be found in the catalog.

Writing Intensive Course Designation

This course has been designated as a "Writing Intensive" (WI) course by City Tech. A WI course includes critical reading, logical thinking, and the use of writing to help students understand the topic; the use of appropriate style and disciplinary conventions in writing and speaking; the use of research resources, including the library, specific to the discipline; a detailed syllabus; a comprehensive course calendar; and a minimum of fifteen pages of writing per student. Writing assignments will be both formal (graded) and informal (non-graded).

Written work will be a mandatory part of the course. This can include coding projects, proofs, and written assignments. Written work will account for a minimum of 10% of the overall grade in the course.

MAT 2540 Discrete Structures and Algorithms II Text: Discrete Mathematics and its Applications,8th edition, by Rosen

Lec.	Discrete Structures and Algorithms II	Homework
1	5.2 Strong Induction and Well-Ordering (354-362)	(P. 362) 3-7
2	5.3 Recursive Definitions and Structural Induction (365-368*)	(P. 378) 1-9 odd, 20*
3	5.4 Recursive Algorithms (381-391)	(P. 391) 1, 3, 5, 7-11, 29-31, 44-48
4	6.3 Permutations and Combinations (428-434) (optional)	(P. 435) 1-7, 9, 11, 17, 19, 33
5	6.1 The Basics of Counting (Skip trees) (405-415)	(P. 416) 1-15 odd, 19, 21, 25, 29, 33, 37, 47, 49, 51
6	6.2 The Pigeonhole Principle (420-426)	(P. 426) 1, 3, 7, 9, 11, 17, 21
7	**Review**	
8	**Test 1**	
9	8.1 Applications of Recurrence Relations (527-536)	(P. 536) 1, 3, 4, 7-9, 11, 13, 21, 26, 27, 28
10-11	8.2 Solving Linear Recurrence Relations (540-550)	(P. 550) 1, 3, 7, 9, 11, 23-25, 27-30
12-13	8.3 Divide-and-Conquer Algorithms and Recurrence Relations (553-561)	(P. 561) 1, 7, 9-17, 21, 36, 37
14	10.1 Graphs and Graph Models (673-682)	(P. 682) 3-9, 29-31, 35
15	10.2 Graph Terminology and Special Types of Graphs (685-699)	(P. 699) 1-10, 20, 21-25, 37, 38-45, 61, 63
16	10.3 Representing Graphs and Graph Isomorphism (703-710)	(P. 710) 1-23 odd, 30, 31, 34, 35, 39-45 odd
17	**Review**	
18	**Test 2**	
19	10.4 Connectivity (skip counting paths) (714-724)	(P. 724) 1-5, 11, 31-34, 50, 54
20	10.5 Euler and Hamilton Paths (728-739)	(P. 739) 1-7 odd, 9, 10, 13-15, 19, 21, 30-36, 37-39, 47
21	10.6 Shortest-Path Problems (743-751)	(P. 751) 2-7, 11, 17, 25, 27
22	11.1 Introduction to Trees (781-791)	(P. 791) 1-9 odd, 17-20, 27, 28
23	11.2 Application of Trees (793-805)	(P. 805) 1-4, 6-8, 11, 19-22, 37
24	11.3 Tree Traversal (808-819)	(P. 819) 1, 3, 6, 7-15, 22-24
25	**Review**	
26	**Test 3**	
27	11.4 Spanning Trees (821-832) 11.5 Minimum Spanning Trees (835-839)	(P. 832) 2-6, 13-18, 27, 28, 30 (P. 839) 1-3, 6, 7
28	5.3 Recursive Definitions and Structural Induction (Recursively Defined Sets and Functions) (370-377)	(P. 378) 22, 23, 25, 34, 35, 36-38, 45, 46
29	**Review**	
30	**Final Exam**	

MAT 2540 Discrete Structures and Algorithms II Text: Discrete Mathematics and its Applications,7th edition, by Rosen

Lec.	Discrete Structures and Algorithms II	Homework
1	5.2 Strong Induction and Well-Ordering (333-341)	(P. 341) 3-7
2	5.3 Recursive Definitions and Structural Induction (344-347*)	(P. 357) 1-9 odd, 20*
3	5.4 Recursive Algorithms (360-370)	(P. 370) 1, 3, 5, 7-11, 29-31, 44-48
4	6.3 Permutations and Combinations (407-413) (optional)	((P. 413) 1-7, 9, 11, 17, 19, 31
5	6.1 The Basics of Counting (Skip trees) (385-394)	(P. 396) 1-15 odd, 19, 21, 25, 29, 33, 37, 45, 47, 49
6	6.2 The Pigeonhole Principle (399-404)	(P. 405) 1-9 odd, 15, 19
7	**Review**	
8	**Test 1**	
9	8.1 Applications of Recurrence Relations (501-505)	(P. 510) 1, 3, 4, 7-9, 11, 13, 21, 26, 27, 28
10-11	8.2 Solving Linear Recurrence Relations (514-524)	(P. 524) 1, 3, 7, 9, 11, 23-25, 27-30
12-13	8.3 Divide-and-Conquer Algorithms and Recurrence Relations (527-534)	(P. 535) 1, 7, 9-17, 21, 36-37
14	10.1 Graphs and Graph Models (641-649)	(P. 649) 3-9, 27-29, 33
15	10.2 Graph Terminology and Special Types of Graphs (651-665)	(P. 665) 1-10, 20, 21-25, 35, 36-43, 59, 61
16	10.3 Representing Graphs and Graph Isomorphism (668-675)	(P. 675) 1-23 odd, 26, 27, 30, 31, 35-41 odd
17	**Review**	
18	**Test 2**	
19	10.4 Connectivity (skip counting paths) (678-689)	(P. 689) 1-5, 11, 31-34, 50, 54
20	10.5 Euler and Hamilton Paths (693-703)	(P. 703) 1-7 odd, 9, 10, 13-15, 19, 21, 30-36, 37-39, 47
21	10.6 Shortest-Path Problems (707-716)	(P. 716) 2-7, 11, 17. 25, 27
22	11.1 Introduction to Trees (745-755)	(P. 755) 1-9 odd, 17-20, 27, 28
23	11.2 Application of Trees (757-769)	(P. 769) 1-4, 6-8, 11, 19-22, 37
24	11.3 Tree Traversal (772-782)	(P. 783) 1, 3, 6, 7-15, 22-24
25	**Review**	
26	**Test 3**	
27	11.4 Spanning Trees 11.5 Minimum Spanning Trees	(P. 795) 2-6, 13-18, 27, 28, 30 (P. 802) 1-3, 6, 7
28	5.3 Recursive Definitions and Structural Induction (Recursively Defined Sets and Functions) (349-356)	(P. 358) 22, 23, 25, 32, 33, 34-36, 43, 44
29	**Review**	
30	**Final Exam**	

(see the next page for a list of suggested projects)

List of Suggested Projects

Lecture 1: Ackermann's Function (Section 5.3, P. 359, Exercise 48-55)

1. Lecture 2: QuickSort Algorithm (Section 5.4, P. 371, Exercise 49-52)
2. Lecture 9: A variation to the Towers of Hanoi Game (Section 8.1, P. 512, Exercise 32)
3. Lecture 9: The Josephus Problem (Section 8.1, P. 512, Exercise 33-37)
4. Lecture 11: Lucas Numbers (Section 8.2, P. 525, Exercise 11)
5. Lecture 15: Coding Graphs (nodes and edges) in a programming language (e.g. Python, C++, or Java)
6. Lecture 22: Coding Family Trees, and Boolean functions checking relationships between family members
7. Lecture 23: Coding Binary Search Trees (e.g. in Python, C++, or Java)
8. Lecture 28: Coding a recursive algorithm that computes the height of any tree; for example, a binary search tree.